Choose Your Own Project

2023-12-05

Introduction

Cardiovascular disease is the leading cause of morbidity and mortality world wide. Of these diseases, myocardial infarctions, better known as heart attacks, is perhaps the most well known. An essential component in reducing the incident of this disease is the accurate identification of a person's risk factor and likelihood for an event.

To better understand the contributing factors in causing heart attacks, in 1948, a large generational study known as the Framingham Heart Study was carried out where a plethora of metrics were measured on the participating individuals. The data for this study was made available and as of now some over 3000 scientific articles have been based on its data.

This project aims to use the Framingham Heart Study data to predict the likelihood of a given individual to get a heart attack, splitting the groups into likely and unlikely (here as factors 0 for unlikely and 1 for likely). The goal in which is the make individuals aware that they may be at risk for a heart attack and to recommend initial lifestyle changes to try and mitigate the risks.

Method/Analysis

First, the data is prepared:

```r
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(riskCommunicator)) install.packages(
  "riskCommunicator", repos = "http://cran.us.r-project.org")
if(!require(party)) install.packages("party", repos = "http://cran.us.r-project.org")
if(!require(kernlab)) install.packages("kernlab", repos = "http://cran.us.r-project.org")

library(riskCommunicator)
library(tidyverse)
library(caret)
library(party)
library(kernlab)

data("framingham")
set.seed(10)
# we aim to identify factors which allow prediction of myocardial infarctions

# data preparation:
select_columns = c('SEX',
                   'TOTCHOL',
                   'AGE',
                   'SYSBP',
                   'DIABP',
                   'BMI',
                   'BPMEDS',
                   'HEARTRTE',
                   'GLUCOSE',
                   'MI_FCHD',
                   'PREVCHD',
                   'PREVSTRK',
                   'PREVMI'
                   )

dataset_cleaned = framingham[select_columns]

dataset_cleaned = dataset_cleaned %>% mutate(MI_FCHD_fact = as.factor(MI_FCHD))
# remove NA data
dataset_cleaned = na.omit(dataset_cleaned)
```

In this case the data is filtered for only a small subset of factors. These factors are data that typically can be examined with a simple blood test or medical history check, this adds relevance in the findings as the model is in theory usable in a walk in clinical setting.

To examine the initial data arrangement:

```
head(dataset_cleaned, 10)
```

```
## # A tibble: 10 x 14
##      SEX TOTCHOL   AGE SYSBP DIABP   BMI BPMEDS HEARTRTE GLUCOSE MI_FCHD PREVCHD
##    <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl>   <dbl>   <dbl>   <dbl>
##  1     1     195    39  106   70    27.0      0       80      77       1       0
##  2     2     250    46  121   81    28.7      0       95      76       0       0
##  3     2     260    52  105   69.5  29.4      0       80      86       0       0
##  4     2     237    58  108   66    28.5      0       80      71       0       0
##  5     1     245    48  128.  80    25.3      0       75      70       0       0
##  6     1     283    54  141   89    25.3      0       75      87       0       0
##  7     2     225    61  150   95    28.6      0       65     103       0       0
##  8     2     232    67  183  109    30.2      0       60      89       0       0
##  9     2     285    46  130   84    23.1      0       85      85       0       0
## 10     2     343    51  109   77    23.5      0       90      72       0       0
## # i 3 more variables: PREVSTRK <dbl>, PREVMI <dbl>, MI_FCHD_fact <fct>
```

```
summary(dataset_cleaned)
```

```
##       SEX           TOTCHOL            AGE            SYSBP      
##  Min.   :1.000   Min.   :112.0   Min.   :32.00   Min.   : 83.5  
##  1st Qu.:1.000   1st Qu.:210.0   1st Qu.:47.00   1st Qu.:120.0  
##  Median :2.000   Median :239.0   Median :54.00   Median :132.0  
##  Mean   :1.555   Mean   :241.7   Mean   :54.29   Mean   :135.8  
##  3rd Qu.:2.000   3rd Qu.:269.0   3rd Qu.:61.00   3rd Qu.:148.9  
##  Max.   :2.000   Max.   :696.0   Max.   :81.00   Max.   :295.0  
##      DIABP             BMI            BPMEDS          HEARTRTE     
##  Min.   : 30.00   Min.   :14.43   Min.   :0.0000   Min.   : 42.00  
##  1st Qu.: 75.00   1st Qu.:23.10   1st Qu.:0.0000   1st Qu.: 69.00  
##  Median : 82.00   Median :25.45   Median :0.0000   Median : 75.00
##  Mean   : 82.95   Mean   :25.84   Mean   :0.0829   Mean   : 76.73
##  3rd Qu.: 90.00   3rd Qu.:28.03   3rd Qu.:0.0000   3rd Qu.: 85.00
##  Max.   :150.00   Max.   :56.80   Max.   :1.0000   Max.   :150.00
##     GLUCOSE          MI_FCHD          PREVCHD           PREVSTRK
##  Min.   : 39.00   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.: 72.00   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median : 79.00   Median :0.0000   Median :0.00000   Median :0.00000
##  Mean   : 83.75   Mean   :0.1539   Mean   :0.06844   Mean   :0.01227
##  3rd Qu.: 89.00   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :478.00   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##      PREVMI        MI_FCHD_fact
##  Min.   :0.00000   0:8134
##  1st Qu.:0.00000   1:1480
##  Median :0.00000
##  Mean   :0.03131
##  3rd Qu.:0.00000
##  Max.   :1.00000
```

There is a mix of discreet factors and continuous variables. Column MI_FCHD indicates whether the person had an infarction event. This column is Of interest as only 1480 of roughly 9500 entries have had an infarction event.

To confirm this the actual proportion can be checked:

```r
# proportion with MI
mean(as.logical(dataset_cleaned$MI_FCHD))
```

```
## [1] 0.1539422
```

Only 15% of the cohort suffered from MI. This adds complexity to the modelling, as simply guessing that a person is not at risk would result in an accuracy of roughly 85% leaving little room for improvement from additional variables. Therefore, the next step is to balance the training data.

```r
# only 15 pct have MI, that means 85% accuracy is achievable by saying everyone is healthy.
# to even the proportions:
MI = dataset_cleaned[dataset_cleaned$MI_FCHD_fact == 1,]
normal = dataset_cleaned[dataset_cleaned$MI_FCHD_fact == 0,]
# sample normals so that it matches MI length
normal_sample = normal[sample(nrow(normal), nrow(MI)), ]
# join the df back together
dataset_cleaned = rbind(MI,normal_sample)

print(mean(as.logical(dataset_cleaned$MI_FCHD)))
```

```
## [1] 0.5
```

Now that the base accuracy is 50% models will now be forced to improve this based on the variables supplied. From this data training and validation data is split.

```r
# split train and test
test_index = createDataPartition(
  y = dataset_cleaned$MI_FCHD_fact, times = 1, p = 0.5, list = FALSE)
train_set = dataset_cleaned[-test_index,]
val_set = dataset_cleaned[test_index,]
```

The general strategy from here is to examine variable ML models to identify what the expected performance is and which model has the highest performance.

Too see which factor likely contributes the most to the prediction functions, an linear model is fitted across all factors.

```r
# test factor dependence using linear modelling:

linear_model = lm(MI_FCHD ~. -MI_FCHD_fact, data = train_set)
summary(linear_model)
```

```
##
## Call:
## lm(formula = MI_FCHD ~ . - MI_FCHD_fact, data = train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.06945 -0.38565 -0.06871  0.42562  0.93772
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.4996695  0.1448211  -3.450 0.000576 ***
## SEX         -0.2610632  0.0243884 -10.704  < 2e-16 ***
## TOTCHOL      0.0017548  0.0002680   6.547 8.09e-11 ***
## AGE          0.0017904  0.0014416   1.242 0.214469
## SYSBP        0.0029665  0.0007869   3.770 0.000170 ***
## DIABP        0.0005614  0.0014417   0.389 0.697020
## BMI          0.0052309  0.0030662   1.706 0.088226 .
## BPMEDS       0.0101504  0.0419650   0.242 0.808909
## HEARTRTE     0.0008699  0.0009443   0.921 0.357120
## GLUCOSE      0.0015513  0.0003798   4.084 4.66e-05 ***
## PREVCHD      0.1854313  0.0494331   3.751 0.000183 ***
## PREVSTRK    -0.0712050  0.0913245  -0.780 0.435698
## PREVMI       0.2583854  0.0618235   4.179 3.10e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4426 on 1467 degrees of freedom
## Multiple R-squared:  0.2234, Adjusted R-squared:  0.2171
## F-statistic: 35.17 on 12 and 1467 DF,  p-value: < 2.2e-16
```

There looks to be very strong dependence on sex, total_cholesterol, glucose and previous heart disease. Interestingly previous infarctions while significant at the 95% confidence, does not seem to be necessarily a very strong predictor for future infarctions.

To test the predictive power of a linear model:

```
lm_prediction = predict(linear_model, val_set)
lm_prediction = as.numeric(lm_prediction > 0.5)
lm_accuracy = mean(lm_prediction == val_set$MI_FCHD_fact)

print(paste0('LM Accuracy:',lm_accuracy))
```

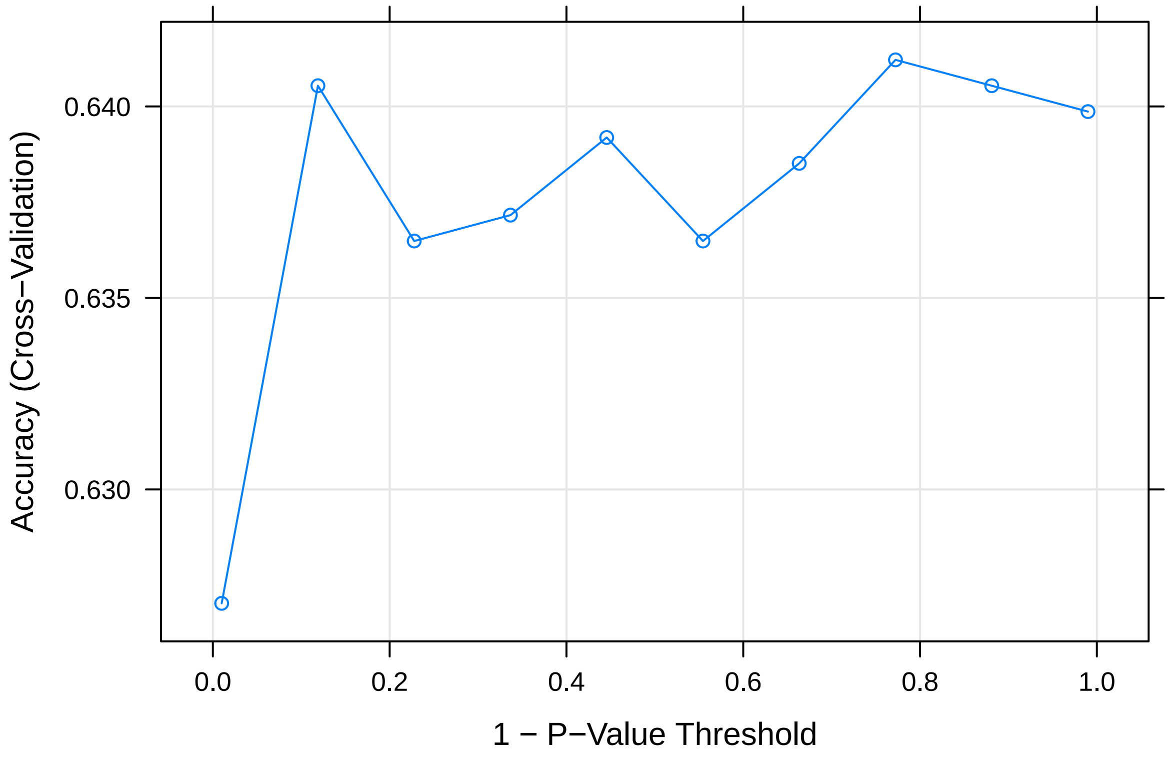
```
## [1] "LM Accuracy:0.706081081081081"
```

About 70.6% predictive accuracy with a multi-factor linear model.

Next a set of additional discriminators are tested. Starting with a decision tree

```
# train a decision tree:
tc = trainControl(method = "cv", number=10)
rt_fit = train(
  MI_FCHD_fact ~. -MI_FCHD,
  data = train_set,
  method = "ctree",
  trControl=tc,
  tuneLength=10)
```

This is then checked with plots of the fitting parameters



```
## Conditional Inference Tree
##
```

```
## 1480 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1332, 1332, 1332, 1332, 1332, 1332, ...
## Resampling results across tuning parameters:
##
##   mincriterion  Accuracy   Kappa
##   0.0100000     0.6270270  0.2540541
##   0.1188889     0.6405405  0.2810811
##   0.2277778     0.6364865  0.2729730
##   0.3366667     0.6371622  0.2743243
##   0.4455556     0.6391892  0.2783784
##   0.5544444     0.6364865  0.2729730
##   0.6633333     0.6385135  0.2770270
##   0.7722222     0.6412162  0.2824324
##   0.8811111     0.6405405  0.2810811
##   0.9900000     0.6398649  0.2797297
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mincriterion = 0.7722222.
```

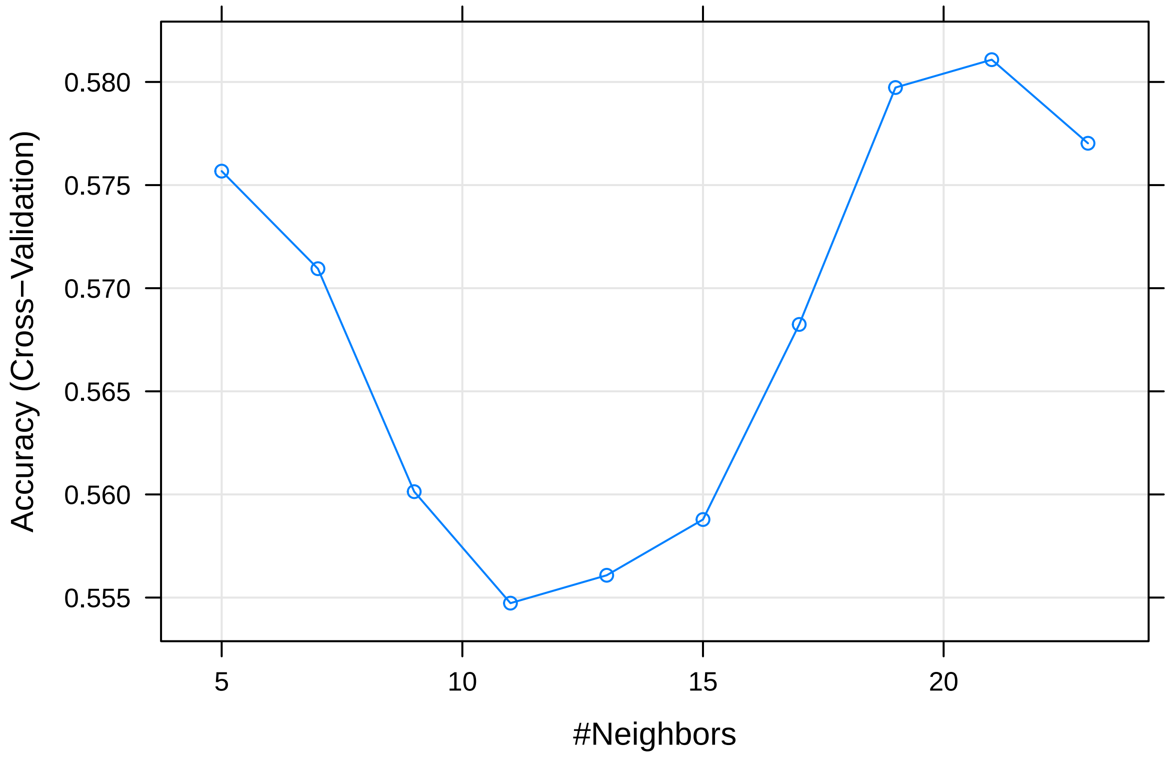
```
## [1] "Descision Tree Accuracy:0.67972972972973"
```

A crossvalidated accuracy of 0.68 was achieved with the mincriterion parameter of 0.7722. This criterion is used for the final fitted model. Cross validation is important as it reduces the risk of overfitting on the training data by scrambling the dataset on each resample.

An additional 4 other classification models is then tested to identify which has the best performance.

KNN:

```
## k-Nearest Neighbors
##
## 1480 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1332, 1332, 1332, 1332, 1332, 1332, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.5756757  0.1513514
##    7  0.5709459  0.1418919
##    9  0.5601351  0.1202703
##   11  0.5547297  0.1094595
##   13  0.5560811  0.1121622
##   15  0.5587838  0.1175676
##   17  0.5682432  0.1364865
##   19  0.5797297  0.1594595
##   21  0.5810811  0.1621622
##   23  0.5770270  0.1540541
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 21.
```



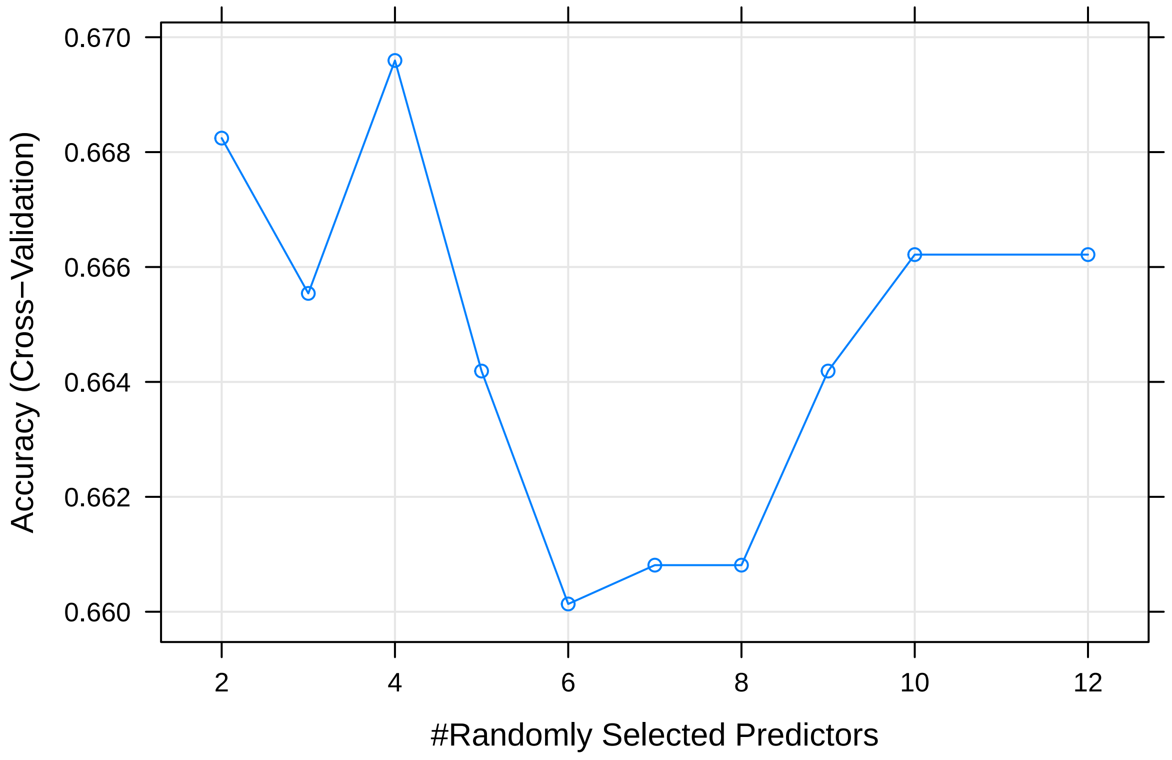
```
## [1] "KNN Accuracy:0.589864864864865"
```

Performance of the KNN model is quite poor in comparison with linear models. Peak accuracy occurs at a large amount of neighbors, but even at that point the predictive accuracy is over 10% less than the linear model.

Random Forest:

```
## Random Forest
##
## 1480 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1184, 1184, 1184, 1184, 1184
## Resampling results across tuning parameters:
##
```

```
##   mtry  Accuracy   Kappa
##    2    0.6682432  0.3364865
##    3    0.6655405  0.3310811
##    4    0.6695946  0.3391892
##    5    0.6641892  0.3283784
##    6    0.6601351  0.3202703
##    7    0.6608108  0.3216216
##    8    0.6608108  0.3216216
##    9    0.6641892  0.3283784
##   10    0.6662162  0.3324324
##   12    0.6662162  0.3324324
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```



```
## [1] "Random Forest Accuracy:0.672972972972973"
```

The results for random forest seems interesting. For the individual trees, there appears to be favoritism towards the extremes of complexity, with either a very small tree (4 predictors) or very large tree (12 predictors) being the most accurate.

Generalized Linear Model:

```
##
## Call:
```

```
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8692  -0.9517  -0.1510   1.0203   2.1675
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.585233   0.790087  -7.069 1.56e-12 ***
## SEX         -1.278480   0.129425  -9.878  < 2e-16 ***
## TOTCHOL      0.009589   0.001465   6.545 5.93e-11 ***
## AGE          0.009234   0.007486   1.234 0.217351
## SYSBP        0.014440   0.004089   3.531 0.000413 ***
## DIABP        0.004593   0.007484   0.614 0.539384
## BMI          0.025789   0.015596   1.654 0.098206 .
## BPMEDS       0.035860   0.215264   0.167 0.867696
## HEARTRTE     0.004560   0.004929   0.925 0.354885
## GLUCOSE      0.010607   0.002552   4.155 3.25e-05 ***
## PREVCHD      0.886412   0.259745   3.413 0.000643 ***
## PREVSTRK    -0.419515   0.493100  -0.851 0.394897
## PREVMI       2.742384   0.576490   4.757 1.96e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2051.7  on 1479  degrees of freedom
## Residual deviance: 1655.9  on 1467  degrees of freedom
## AIC: 1681.9
##
## Number of Fisher Scoring iterations: 6
```

```
## [1] "Generalized Linear Model Accuracy:0.704729729729730"
```

GLM shows performance similar to standard linear modelling.

The overall accuracy for all models sits somewhere between 60 and 70%. A method to possibly further improve the results is to use an ensemble where multiple models for training data is used and the final value is chosen on consensus. To implement a simple version of this, a voting system is introduced with majority votes resulting in the selected outcome.

As an ensemble:

```r
# use the model predictions to create an ensemble
# will have a voting system where majority rules
ct_vote = as.numeric(ctree_prediction)-1
knn_vote = as.numeric(knn_prediction)-1
rf_vote = as.numeric(rf_prediction)-1
glm_vote = as.numeric(glm_prediction)-1
lm_vote = as.numeric(lm_prediction)-1

vote_tally = ct_vote+knn_vote+rf_vote+glm_vote+lm_vote
final_prediction = as.factor(as.numeric(vote_tally >= 3))
accuracy = mean(final_prediction == val_set$MI_FCHD_fact)
print(accuracy)
```

```
## [1] 0.6824324
```

Interestingly the ensemble prediction accuracy is lower than that for the GLM and LM. suggesting the classifier models may be dragging down the accuracy.

Results and discussion

The final results obtained from this exercise is as follows:

```
## [1] "LM Accuracy:0.706081081081081"

## [1] "Decision Tree Accuracy:0.67972972972973"

## [1] "KNN Accuracy:0.589864864864865"

## [1] "Random Forest Accuracy:0.672972972972973"

## [1] "Generalized Linear Model Accuracy:0.70472972972973"

## [1] "Ensemble Accuracy:0.682432432432432"
```

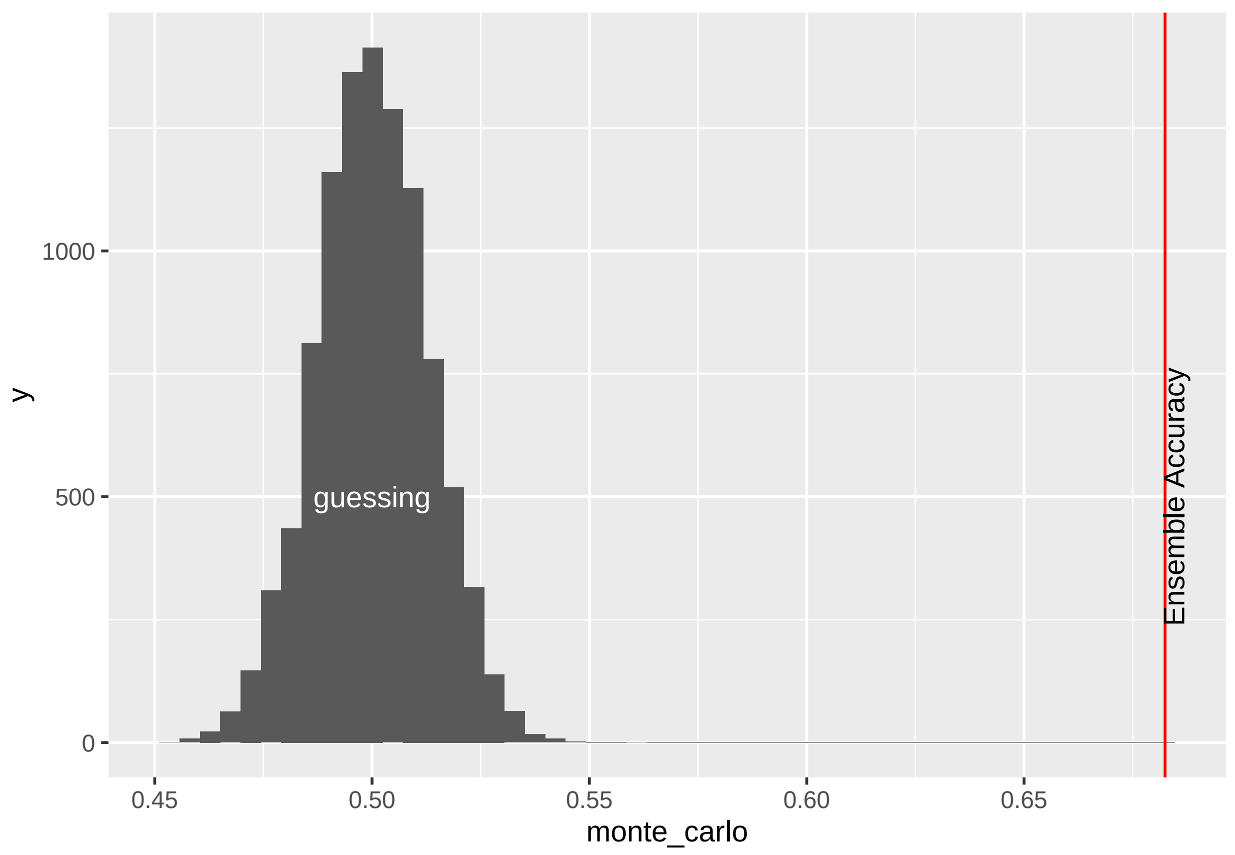
From these results we find that the two linear models (LM and GLM) performs best at this task, while KNN is worst. However, interestingly, predictive accuracy for all the models were comparatively low. If such a system is used for advising people of their risk factor for MI occurring, then ideally the performance would be higher.

In terms of the low predictive accuracy, it's likely that the current accuracy is close to the maximum accuracy achievable with the set of predictors selected. This is noted as most models predict similar accuracy levels and an ensemble of the models did not increase predictive power. In terms of the underlying mechanism of infarctions, there is potentially extensive random factors and additional factors not measured in the overall study from which the data is derived.

To check if the predicted accuracies is beyond what is likely by chance, a monte carlo simulation can be carried out to see the expected results from randomly guessing results:

```r
guessing = function (g){
  random_guess = rbinom(nrow(val_set), 1, 0.5)
  random_guess_accuracy = mean(final_prediction == as.factor(random_guess))
}

monte_carlo = sapply(1:10000, guessing)
ggplot() + geom_histogram(aes(x = monte_carlo ), bins=50) +
  geom_vline(aes(xintercept = accuracy, ), color='red') +
  annotate("text", x=accuracy+0.002, y=500, label="Ensemble Accuracy", angle=90) +
  annotate("text", x=0.5, y=500, label="guessing", color = 'white')
```



we note in comparison to the monte carlo simulated guesses, the models perform considerable better and so offers some predictive ability over random guessing.

Conclusion

This study aimed to investigate the ability to develop a method for predicting a person's susceptibility to myocardial infarctions through the use of various machine learning models. We noted that the 5 methods tested all produced results sitting in the range of 60 to 70 percent accuracy. Testing against random generated numbers howed that while low, these predicted values were higher than what is expected from random sampling, proving that there is predictive power present within the model.

The ability to accurately predict onset of infarctions is essential to reducing the burden of disease from heart disease and so pursuits into this kind of predictive modelling is of significant impact if it is able to do so with high degrees of accuracy.

Future work in this field will focus on discovering additional predictors to better improve the accuracy of the predictions. In addition to this, instead of predicting on a binary scale of 'at risk' and 'not at risk' a sliding scale should be implemented where the predicted value is a range, stretching between these two extremes.

Overall, the results of this project is a start in modelling for prediction of infarctions.

References

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D. A., François, R., … & Yutani, H. (2019). Welcome to the Tidyverse. Journal of open source software, 4(43), 1686.

Grembi, J. A., & Rogawski McQuade, E. T. (2022). Introducing riskCommunicator: An R package to obtain interpretable effect estimates for public health. PLoS One, 17(7), e0265368.

Kuhn, M. (2015). Caret: classification and regression training. Astrophysics Source Code Library, ascl-1505.

Hothorn, T., Hornik, K., Strobl, C., Zeileis, A., & Hothorn, M. T. (2015). Package ‘party’. Package Reference Manual for Party Version 0.9-998, 16, 37.

Karatzoglou, A., Smola, A., Hornik, K., & Zeileis, A. (2004). kernlab-an S4 package for kernel methods in R. Journal of statistical software, 11, 1-20.